ARL-CR-0779 ● SEP 2015

US Army Research Laboratory

High-Bandwidth Tactical-Network Data Analysis in a High-Performance-Computing (HPC) Environment: Packet-Level Analysis

prepared by Brian Panneton
Technical and Project Engineering, LLC
Alexandria, VA

James Adametz
QED Systems, LLC
Aberdeen, MD

under contract W91CRB-11-D-0007

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-CR-0779 ● SEP 2015

US Army Research Laboratory

High-Bandwidth Tactical-Network Data Analysis in a High-Performance-Computing (HPC) Environment: Packet-Level Analysis

prepared by Brian Panneton
Technical and Project Engineering, LLC
Alexandria, VA

James Adametz
QED Systems, LLC
Aberdeen, MD

under contract W91CRB-11-D-0007

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
September 2015	Final	July 2012–December 2014

4. TITLE AND SUBTITLE
High-Bandwidth Tactical-Network Data Analysis in a High-Performance-Computing (HPC) Environment: Packet-Level Analysis

5a. CONTRACT NUMBER
W91CRB-11-D-0007

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Brian Panneton and James Adametz

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Technical and Project Engineering, LLC
Alexandria, VA
QED Systems, LLC
Aberdeen, MD

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
US Army Research Laboratory
ATTN: RDRL-CIH-C
Aberdeen Proving Ground, MD 21005

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
ARL-CR-0779

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This report describes how the collaboration of Army Test and Evaluation Command's Aberdeen Test Center and the US Army Research Laboratory enabled a system to be built that leverages the capabilities of the high-performance-computing systems to produce a data model used to analyze tactical radio network performance at the packet level.

15. SUBJECT TERMS
tactical networks, data reduction, high-performance computing, data analysis, big data

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Kenneth Renard
Unclassified	Unclassified	Unclassified	UU	28	19b. TELEPHONE NUMBER (Include area code) 410-278-4678

Contents

List of Figures

List of Tables

INTENTIONALY LEFT BLANK.

1. Introduction

Tactical radios are held to more stringent requirements than radios found in the commercial field. They must adhere to a higher set of requirements, which allow them to better operate in hostile environments. The Brigade Modernization Command, in conjunction with the Army Test and Evaluation Command's Operational Test Command, conducted large-scale test events such as the Network Integration Evaluation[1] to test radios in relevant tactical environments. Radio networks under test are instrumented to record traffic transmitted between network nodes. These data are processed and analyzed to determine how well a single radio or the whole network performed in the test.

The collaboration between the Aberdeen Test Center and the US Army Research Laboratory's Computational and Information Sciences Directorate resulted in a data processing system that reduces the collected traffic into manageable data products. Analysts extract relevant metrics from these data products to support the evaluation of the system-under-test performance.

One of these data products, *CommsIp*, is related to packet-level analysis and is critical to network evaluation. This data product includes statistics such as the latency and completion rates between network nodes; both are derived by correlating the data (packets) observed at each node during the test. Correlating over 1 billion packets recorded during each day of testing was a forcing factor to employ high-performance-computing (HPC) assets to process the massive volumes of data into a usable data model.[2] This report explains the need of this data model as well as the cut module[3] within the HPC framework[4] that creates it.

2. Motivation and Desired Outputs

Most systems send data across a network encapsulated in an IP packet. The IP layer data may arrive successfully, arrive out of order, or be dropped during transit. The results of these 3 cases are needed to determine several aspects of network performance.

IP is known as a "best effort" delivery protocol, and knowing how well the network delivers IP layer data provides insights into how well the network performs from the end user perspective. By performing IP layer packet analysis, one can determine key network performance metrics, such as data delivery latency, load handling, and overall delivery completion rates. This analysis becomes even more important when the network is in the tactical domain because of the need for reliable and secure networking.

To evaluate network performance, the CommsIp data model is generated from raw collected data. The model captures the history of each packet and includes details from each end point that it was observed at.

During the process of creating the CommsIp data model, it is possible to perform calculations that analysts commonly want to see in a distributed parallel fashion that can significantly reduce the time it takes to achieve the same result in a serially processed, database-driven analysis. This includes packet matching, latency calculation, packet endpoint determination, and the filtering out of local network traffic, which is of no interest from an analysis perspective.

The Test and Evaluation community has determined most of the definitions and layout of the data product, which can be seen in Table A-1 (see the Appendix). Extra columns (refer to Table A-2) have been added to reduce unneeded data reduction for other cut modules, such as Transport, which draws Transmission Control Protocol (TCP)-based statistics. Each row in the CommsIp table typically represents the combination of 2 observations of a packet, the sending side and the receiving side. In some cases only one side of the transmission will be observed and the row will reflect that by leaving fields empty that cannot be calculated without both observations. An example would be calculating latency where you must have both sides.

3. Data Organization

This section describes in detail what reductions and manipulations occur within the CommsIp cut module. The module takes 2 different types of input data cuts. The cuts come from Binary Large Object (BLOb) files and/or Packet Capture (PCAP) files. During the module's Process stage, important information is pulled from packets and saved in a temporary data store. This information is then read in during the module's Crunch stage, where packet matching calculations on the data occur. This simplified data are then turned into the CommsIp Data Product.

4. Prepare File

File metadata must be collected to properly organize the information found in data cuts. The device ID, file ID, and, in the case of a PCAP file, the recording source are all required for correlating packet data across nodes in a network.

The device ID is an identifier used in mapping an Advanced Distributed Modular Acquisition System (ADMAS) to its recorded data. The file metadata provides a serial number that is used to look up the device ID in a predefined reference.

The file ID is a global identifier that represents the file. Each HPC processing core is able to perform a lookup on a file ID and uniquely access the same file.

5. CommsIp Process

As the file parser begins iterating cuts, they are passed into CommsIp's Process method. Though BLOb Nettap cuts[5] and PCAP cuts are roughly the same, BLOb cuts contain more cut metadata. The metadata for each Nettap cut contains the network source data stream it was recorded from, in comparison with the PCAP whose data are from only one network source. BLOb Nettap cut metadata also contains information about whether or not the data collection device experienced an overflow error (resulting in unrecorded data), what type of overflow error it was, and which interface it was recorded on. If an overflow error occurs, the data in the cut may be corrupted, and for this reason, the cut is ignored.

When the CommsIp cut module receives a cut, it decodes the cut's payload to extract the full Ethernet packet contained within and checks to verify that the packet should be processed. One check compares the packet's collection time to the evaluation time window specified in the user-set configuration file. Another check ensures the packet was collected from a known source and on a tap being considered in the data reduction. Only packets that pass all checks are considered for the remainder of the reduction process.

From here, each verified Ethernet packet has its EtherType[6] decoded. All packets with EtherTypes that are not IPV4 get dropped because of prior knowledge that the evaluations will only be performed on IPV4 data. Tunneled packets are then broken down into their inner and outer IP layers by decoding any tunnel protocols, such as the Generic Routing Encapsulation protocol that may be encapsulating the packet.

Packets with the outer IP layer's time to live (TTL) less than 2 get dropped because of the location on the network of where the ADMAS records traffic. The packet's TTL will typically drop by 2 or more when traveling over the air. Thus, this data may get recorded by the ADMAS but will get dropped before reaching the destination device.

6. Collection Point

The collection point is the location where the ADMAS is observing traffic on the network. There can be many collection points on one node; each is given a letter designation. For example, the collection point "X" is located on the switch port analyzer (SPAN) port of the router facing the over-the-air (OTA) radio. Thus, any traffic coming in or out of the OTA radio will end up being copied to the ADMAS.

The location of the collection point is vital to the analysis. For instance, if one side of the network collects data behind an encrypter and the other side collector is before the encrypter, there is absolutely no way to match the packets.

7. Direction

The location of the collection point is an important part of creating heuristics to determine the direction of packet transmission. Each collection point may use a different heuristic, and the heuristic may change from event to event. Some of the simple cases rely on prior knowledge of the testing setup to determine the direction. More complicated cases involve an extra processing phase to gather more information to make the determination.

For example, collection point "B" as used in most of the Warfighter Information Network–Tactical test events was a simple heuristic using a simple check on the media access control (MAC) addresses in a packet's Ethernet header to determine if the packet is inbound or outbound. The collection point "B" heuristic states that if the packet's source MAC address is an inline network encrypter MAC address, it is considered inbound. Alternately, if the destination MAC address is either multicast or the network encrypter, then it is outbound.

Some of the heuristics will use Virtual Local Area Network identifiers and MAC addresses to determine direction. There are many other heuristics that change from event to event, but they will not be covered in this report.

8. Detecting Local Traffic

Local packets are those that transit from one device to another on the same network node. An example of local network traffic would be a vehicular router pinging a collocated radio device to see if it responds.

A packet can be either transmitting between nodes or transmitting entirely within a single node. Since the CommsIp Data Model strictly contains packets that transmit between nodes, local traffic must be filtered out. Packets that have a source and destination on the same node or a packet direction that cannot be determined are assumed local and removed from processing. This check is sometimes used to determine the packet direction; however, it is generally used during the Crunch stage of the reduction.

9. Hash

To conduct efficient packet matching and calculate the latency, there needs to be a common key between packets recorded on different devices. To generate this common key, a packet's mutable fields are removed since they can change between the sending side and the receiving side. Next, a hashing algorithm is applied to the modified packet resulting in a common key. Mutable fields include the IP options, the TTL, the packet checksum, and the type of service. In addition, the outer IP layer (if a tier-2 tunneling protocol is used) may be completely different because of how the packet gets routed through the network, hopping between tunnel endpoints. By omitting these mutable fields from the packet and only hashing on the inner IP layer, we find that the sending side hash matches the receiving side hash.

Figure 1 depicts in red the fields that are not included in the hash because these fields are mutable. Each hash is associated with the data for its packet stored in the Packet-Knowledge-Temporary-Store (PKTS) local to the CommsIp Worker process. The PKTS is a temporary storehouse for all reduced packet data. PKTS records are incrementally populated during the various phases of the reduction processing. Each process records data into a separate store. The data columns of the store are shown in Tables A-3 and A-4 in the Appendix.

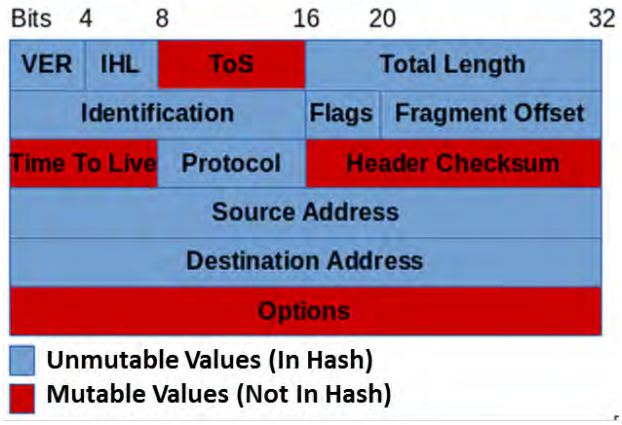

Fig. 1 IP header hashed fields

10. Fragments

Before the PKTS data are recorded, all IP packet fragments are reconstructed.[7] This is done because fragmentation can occur anywhere along the network path and thus may change how the packet appears on the receiving side, making it difficult to match individual fragments using the hash-based method. In general, fragments

appear in order and relatively close to each other in the file. A fragment bookkeeping mechanism is used to collect the fragments.

As the bookkeeper collects packet fragments from a file, it attempts to reconstruct the whole packets they originated from. Fragments from a file that do not fully recreate a packet are provided to the Receiver[8] process to be matched to fragments from other files. Packets that are reconstructed are decoded and have their information added to the PKTS.

11. CommsIp Crunch

Once the Process stage is complete and all fragmented packets have been decoded, the Crunch stage begins. At the start of the CommsIp Crunch stage, the Receiver process begins to offload work to CommsIp workers. Dividing up the work across the processes is crucial to efficient processing of the data. The initial breakdown of the work is based on bins that each packet is placed into. The bins are numerically defined based on the number of processes (P) in the HPC job, where N is determined by Eq. 1,

$$N = ceiling(log_2(P)) \,, \tag{1}$$

and the bins are defined to include the range $0 \longrightarrow 2^N - 1$. As an example, if we had 250 HPC processes for a reduction job, then $N = ceiling(log_2(250)) = 8$, and the bins would be $0 \longrightarrow 255$.

Based on values calculated in the CommsIp Process stage, bin keys are sent to each CommsIp worker. Bin keys are unique bit strings that map to the tailing N bits in packet hashes. The keys identify which set of packets each worker should operate on.

Considerations must be observed for memory for these packet operations since some of the HPC machines do not have swap space.[9] When too much runtime memory is consumed, the node will end the reduction job prematurely. To prevent this, the number of packets per hash—thus, the number of packets that can be processed by each worker at a time—is limited based on the available memory. For current systems, the upper limit is set to 200,000 packets.[10] Though this limit may seem low, it allows larger hash bins to be split up and processed in parallel sub-bins.

During the Process stage, CommsIp records a count of packets per bin (PPB), which allows it to determine when the limit is exceeded by any bin. When the upper limit is exceeded, crunch creates a number of sub-bins (S) according to Eq. 2:

$$S = 2^{ceiling(log_2\left(\frac{PPB}{200{,}000}\right))} . \quad (2)$$

This will normally produce an evenly distributed amount of sub-bins, each being smaller than the upper bound. The CommsIp Crunch process will offload each of these to a worker and then wait until all the processing is done.

12. Offloaded Worker

An offloaded worker is one that receives a bin key, the number of sub-bin bits used, and a sub-bin value. These workers collect all packet data from the combined set of all PKTS that matches the bin key and the sub-bin value.

13. Packet Matching

Once all of the hashes have been found, the offloaded worker must attempt to find sent and received packet matches. To simplify this, while pulling the sub-bin into memory, each packet's data are placed in an indexed table with the packet hash as the key. Each index (hash) can have one or more packet records, so iterating through the table provides a collection of packets with the same hash identifier.

Indexes with only one packet represent an unmatched packet. For these, the packet's direction flag *(pkt_isoutbound* in PKTS), which shows the packet's direction, is examined to determine if this is a *received but not sent* (RNS cip_rns = true) or *not completed* (cip_comp = false) packet.

When more than one packet has the same hash value, then the set of packets is split into two lists (SENT and RECEIVED) based on each packet's direction flag. These lists are then fed into 1 of 2 packet-matching processing algorithms: unicast or multicast.

14. Unicast

The unicast algorithm is outlined in Fig. 2. Unicast matching begins with the 2 packet lists: one for sent packets and one for received packets.

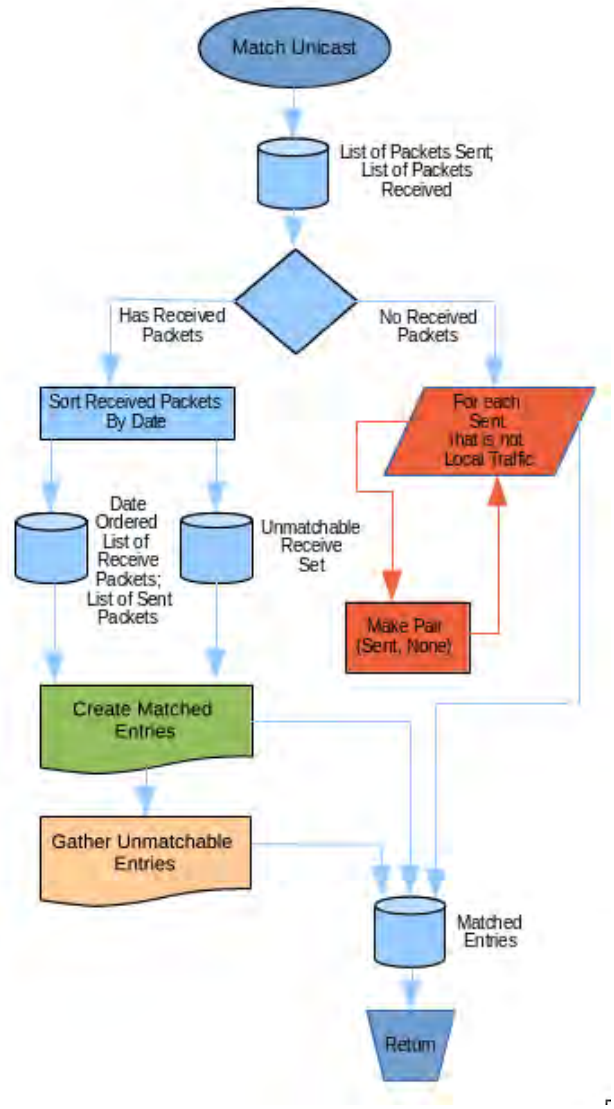

Fig. 2 Unicast packet matching process

The red loop depicts what happens if there are no received packets. Each sent packet is made into a one-sided pair. The pair is then appended to a list of matched entries.

If instead there are received packets, the list of receive-side packets is sorted in ascending order by collection time. Initially, all packets in this list are considered to be “unmatchable”. Then, as the algorithm finds potential matches of sent and received packets, the received packets are removed from the “unmatchable” set. Figure 3 depicts how matches are created from these 3 lists.

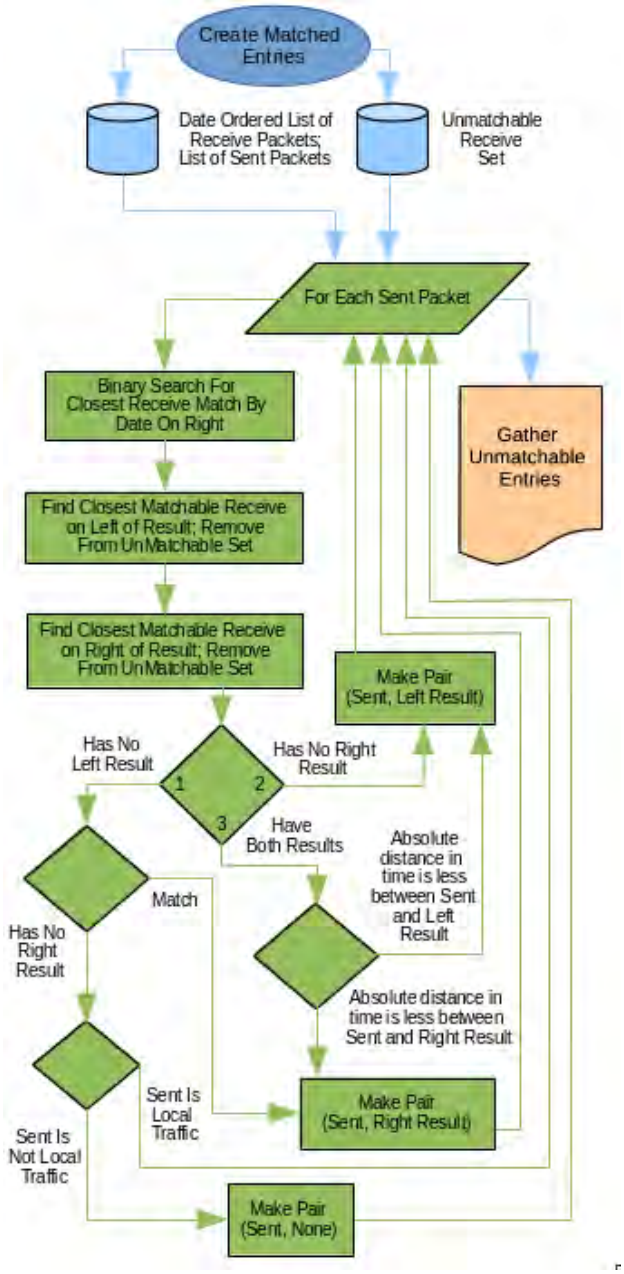

Fig. 3 Creating matched packet entries

For each of the sent packets in the list, a binary search is used to find the received packet that was collected the soonest after and the one that was collected the latest before the sent packet. Any packets found are considered potential matches and are removed from the “unmatchable” set.

The results of the binary search can produce 3 different scenarios as depicted by the first diamond shape in Fig. 3.

1) There are no matchable receivers to the left (earlier in time) side of the binary search result.

2) There are no matchable receivers to the right (later in time) side of the binary search result.

3) There are results on both sides.

In case 1, the algorithm first checks if the right side had a result. If so, a matching pair is made from the sent packet and the right result. If not, the sent packet must be tested to determine if it is local traffic. If the sent packet is considered local, then it is ignored, and the next sent packet is processed. Otherwise, a receiver-less pair (*cip_comp* = false) is created and appended to the list of matched entries.

In case 2, there is a left result but no right result. A pair is made using the sent packet and the left result. The match is added to the list of matched entries.

In case 3, there is a result on the left side and a result on the right side. The algorithm decides which side has the absolute minimum time difference and generates a matched pair with the sent packet and the closer received packet.

Once the algorithm has attempted to match all of the sent packets, there may be some received packets that remain in the “unmatchable” set. The remaining set of unmatchable received packets is converted into receive-side-only pairs, as shown in the tan loop in Fig. 4. The resulting received but not sent (*cip_rns* = true) entries are added to the list of matched entries.

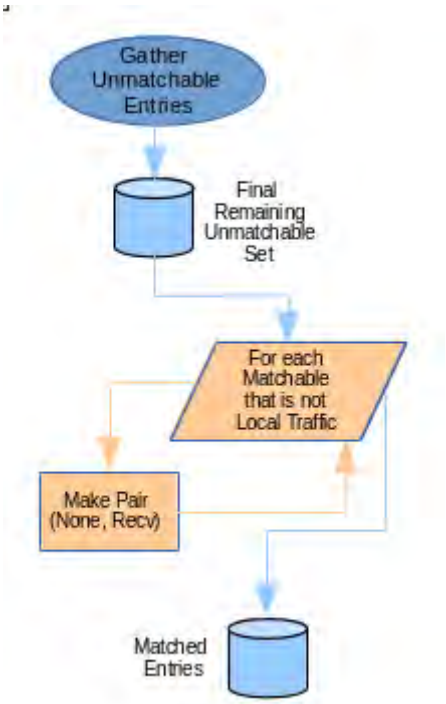

Fig. 4 Handling unmatched packets

15. Multicast

Dealing with a packet hash set that is multicast is similar to unicast, but the algorithm must be taken into account—one sent packet can have multiple receivers. In the unicast case, a packet could only be matched once. In the multicast case, however, it may be matched once for each device observing a received copy. The matching algorithm is modified such that the input list of date-ordered received packets is grouped per device. Then the list of sent packets is traversed in the same manner as in the unicast algorithm but for each device found in the receive list. In addition, the set of unmatchable receives is split up by the device. Aside from this difference, the algorithm works in the same way.

16. Offloaded Worker (Continuation)

Once all of the matches have been found, they are filtered for duplicate observations. Duplicate observations are defined as observations that occur within .000245 s^{11} of a bitwise identical packet on the same device. After the duplicate observations are removed, the matched packets are then merged into the previously described CommsIp table format. The latency calculation is performed during the merge of the 2 packets. In addition, specific flags will also get set, such as cip_ismulticast and cip_isduplicatepkt. The merged data are then written to disk.

17. Analyst Usage

The packet-level data contained on the CommsIp table is used by the analytical community to render many different types of data products. These include aggregate statistics binned by time and/or location within the network. The types of tactical applications or network devices can be derived using the IP addresses contained in each CommsIp record. One sample data product derived from the data model is shown in Fig. 5, a Google Earth[12] Keyhole Markup Language (KML) file. This product includes aggregate network statistics between node pairs. The white lines represent the range between nodes and are used to render the terrain profile between nodes (seen at the bottom of the image). The blue arcs represent satellite communication links between node pairs; green arcs represent terrestrial radio links. The CommsIp-derived data for each link can be displayed by clicking on the links (white box pop-up in upper left of the map area).

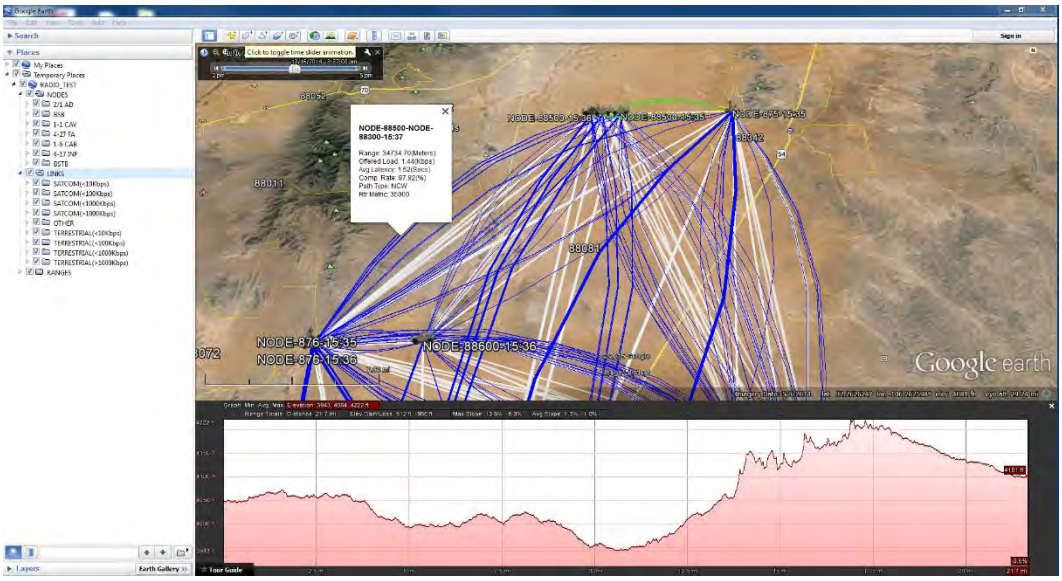

Fig. 5 Sample CommsIp-derived data product (KML file)

18. Conclusion

The packet-level analysis data processing module, CommsIp, has been used for multiple testing events. As the tests evolved, the module has evolved as well to cover new cases and new collection points.

The CommsIp processing represents the bulk of the computational work required to perform packet-level network analysis and requires significant amounts of processing time to complete. However, with the parallel nature of the data reduction framework and HPC machines, the impact on time is mitigated to a reasonable

level. The shortening of the reduction time line allows the analysts to receive this useful data product much faster. Since most of the analysis comes from the CommsIp data product, having it in hand early can allow them to determine the results of the test much faster.

19. References and Notes

1. OT&E Director. DOT&E FY2013 annual report; 2014 Jan 27 [accessed 2015 Aug 11]. http://www.dote.osd.mil/pub/reports/FY2013/pdf/army/2013nie.pdf.

2. Army Aberdeen Test Center (US). C4 data model description document 1.8.13. Aberdeen Proving Ground (MD): Army Aberdeen Test Center (US); 2014.

3. A *cut* as used herein is a record of raw recorded data that comes in many types. Of primary concern to this report is the *nettap* cut type, which contains recorded Ethernet data. A *cut module* is a reduction application component that consumes cuts and renders them into data model records.

4. Panneton B, Adametz J. High-bandwidth tactical-network analysis in a high-performance-computing (HPC) environment: HPC data reduction framework. Aberdeen Proving Ground (MD): Army Research Laboratory (US); 2015 Sep. Report No. ARL-CR-0777.

5. Army Aberdeen Test Center (US). VISION BLOb description. Aberdeen Proving Ground (MD): Army Aberdeen Test Center (US); 2014.

6. IEEE Std. 802.3. IEEE standard for Ethernet. Piscataway (NJ): Institute of Electrical and Electronics Engineers. 2012 Dec 28 [accessed 2014 Aug 2]. http://standards.ieee.org/findstds/standard/802.3-2012.html.

7. Kozierok CM. IP message fragmentation process. 2015 Jan 8 [accessed 2015 Jan]. http://www.tcpipguide.com/free/t_IPMessageFragmentationProcess.htm.

8. Panneton's report *High-Bandwidth Tactical-Network Analysis in a High-Performance-Computing (HPC) Environment: HPC Data Reduction Framework* (see Ref. 3 listed above) contains definitions of the Worker and Receiver processes within the HPC framework.

9. Swap space is a mechanism employed to use more memory than a computer actually has in its RAM storage.

10. 200,000 packets as the upper limit of bin size was empirically determined to be a good value.

11. This number was empirically determined to work well for duplicate observations captured from the CISCO SPAN port. A large number of these duplicate observations was collected and analyzed in early testing to determine what this value should be set to.

12. Google. Google Earth [accessed 2015 July 22]. https://www.google.com/earth/.

Appendix. Tabular Data Definitions

Table A-1 CommsIp data elements

Column Name	Description
cip_xdate	Datetime on transmitting side of when packet was observed
cip_rdate	Datatime on receiving side of when packet was observed
cip_xcollpt	Data collection point on transmitting side
cip_rcollpt	Data collection point on receiving side
cip_comp	Boolean: True if the packet was observed on the receiving side, False otherwise
cip_rns	Boolean: True if the packet was observed on the receiving side but not the sending side, False otherwise
cip_totalpacketsize	Total size of the packet in bytes
cip_payloadsize	Size of the packets inner most payload
cip_latency	Latency between transmission and receipt of packet
cip_xdscp	Differentiated services code point on transmitting side
cip_rdscp	Differentiated services code point on receiving side
cip_xdeid	Device ID on transmitting side
cip_xip	IP Address on transmitting side
cip_rdeid	Device ID on receiving side
cip_rip	IP Address on receiving side
cip_protocol	The protocol the packet was sent on
cip_payloadhash	The folded md5sum hash of the inner most payload (backwards compatibility)
cip_ipidentifier	The IP identifier on the packet that was sent
cip_fragmented	Boolean: True if the packet was fragmented, False otherwise
cip_xttl	The time-to-live value on the transmitting side of the packet
cip_rttl	The time-to-live value on the receiving side of the packet
cip_xsrcmac	The source MAC address on the transmitting side
cip_rsrcmac	The source MAC address on the receiving side
cip_xdstmac	The destination MAC address on the transmitting side
cip_rdstmac	The destination MAC address on the receiving side
cip_t_xip	The outer most tunnel IP address of the transmitter
cip_t_rip	The outer most tunnel IP address of the receiver
cip_innerfingerprintid	The identifying full md5sum hash of the altered inner most IP layer, used for matching
cip_inferred_x_deid	Intended device the packet was sent from
cip_inferred_r_deid	Intended device the packet was destined for
cip_daglimiteduseid	Used for marking out data that should not be used by analysts
cip_dagreasoncodeid	Used for marking out data that should not be used by analysts
cip_ismulticast	Boolean: True if the packet is a multicast packet, False otherwise
cip_isduplicatepkt	Boolean: True if the packet was observed at more than 2 locations, False otherwise
cip_xvlanid	Virtual Local Area Network ID of the sending side packet
cip_rvlanid	Virtual Local Area Network ID of the receiving side packet
cip_istunneled	Boolean: True if the packet is tunneled, False otherwise
cip_t_ipid	The tunnel layers IP identifier of the packet
cip_t_payloadsize	The tunnel layers payload size (contains the inner IP layer)
cip_t_payloadhash	The tunnel layers md5sum payload hash

Table A-2 CommsIp additional transport elements

Column Name	Description
txp_id	13-byte transport identifier string
txp_hash	Hash of "normalized" transport identifier
txp_xport	TCP/UDP port of packet transmitter
txp_rport	TCP/UDP port of packet receiver
txp_istcp	Boolean: True if packet is TCP, false otherwise
tcp_tcp_seq	The sequence number from the TCP header (undefined if not TCP)
txp_tcp_ack	The acknowledgement number from the TCP header (undefined if not TCP)
txp_datalen	The total size of the TCP/UDP payload
txp_tcp_flags	The TCP flags field (undefined if not TCP)
pay_fileid	The internal file ID of the file containing the payload
pay_offset	The file offset location of the payload in the file
pay_length	The length of the payload in bytes

Table A-3 Packet knowledge temporary store structure

Column Name	Description
binkey	The first n bits of the pkt_fingerprint used to bin the data
nexteight	The next 8 bits of the pkt_fingerprint used to sub-bin
pkt_date	The datetime of when the packet was observed
pkt_collpt	Data collection point of the observed packet
pkt_isoutbound	Boolean: True if the packet is outbound, False otherwise
pkt_totalpacketsize	Total size of the packet in bytes
pkt_payloadsize	Size of the packets inner most payload
pkt_dscp	Differentiated services code point of packet
pkt_device	Device ID of the observing ADMAS
pkt_xip	IP Address of the transmitter
pkt_rip	IP Address of the receiver
pkt_protocol	The protocol the packet was sent on
pkt_payloadhash	The folded md5sum hash of the inner most payload (backwards compatibility)
pkt_ipid	IP identifier of the packet
pkt_isfragmented	Boolean: True if packet was fragmented, False otherwise
pkt_packetcount	The number of fragmented packets that the original packet is comprised of
pkt_srcmac	The source MAC address of the packet
pkt_dstmac	The destination MAC address of the packet
pkt_fingerprint	The identifying full md5sum hash of the altered inner most IP layer, used for matching
pkt_ttl	The time-to-live value of the packet
pkt_vlanid	The virtual local area network ID of the packet
pkt_istunneled	Boolean: True if the packet is tunneled, False otherwise
pkt_t_ipid	The IP Identifier of the outer most tunnel layer
pkt_t_xip	The sending IP Address of the outer most tunnel layer
pkt_t_rip	The receiving IP Address of the outer most tunnel layer
pkt_t_payloadsize	The size of the outermost tunnel layers payload (includes the inner IP layer)
pkt_t_payloadhash	The folded md5sum hash of the outer most tunnel payload (backwards compatibility)
pkt_t_fingerprint	The identifying full md5sum hash of the altered outer most IP layer, used for matching
pkt_inferred_x_deid	Intended device the packet was sent from
pkt_inferred_r_deid	Intended device the packet was destined for

Table A-4 Packet-Knowledge-Temporary-Store (PKTS) added transport level fields

Column Name	Description
txp_id	13-byte transport identifier string
txp_hash	Hash of "normalized" transport identifier
txp_xport	TCP/UDP port of packet transmitter
txp_rport	TCP/UDP port of packet receiver
txp_istcp	Boolean: True if packet is TCP, false otherwise
tcp_tcp_seq	The sequence number from the TCP header (undefined if not TCP)
txp_tcp_ack	The acknowledgment number from the TCP header (undefined if not TCP)
txp_datalen	The total size of the TCP/UDP payload
txp_tcp_flags	The TCP flags field (undefined if not TCP)
pay_fileid	The internal file ID of the file containing the payload
pay_offset	The file offset location of the payload in the file
pay_length	The length of the payload in bytes

List of Symbols, Abbreviations, and Acronyms

ADMAS	Advanced Distributed Modular Acquisition System
BLOb	binary large object
FPGA	field-programmable gate array
HPC	high-performance computing
IP	Internet Protocol
KML	Keyhole Markup Language
MAC	media access control
OTA	over-the-air [radio]
PCAP	Packet Capture
PKTS	Packet-Knowledge-Temporary-Store
TTL	time to live

1 (PDF) DEFENSE TECHNICAL INFORMATION CTR DTIC OCA

2 (PDF) DIRECTOR US ARMY RESEARCH LAB RDRL CIO LL IMAL HRA MAIL & RECORDS MGMT

1 (PDF) GOVT PRINTG OFC A MALHOTRA

1 (PDF) TECH AND PROJ ENGR LLC B PANNETON

1 (PDF) QED SYSTEMS LLC J ADAMETZ

1 (PDF) DIR USARL RDRL CIN S K RENARD